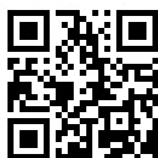
RAZzies

Maandblad van de Radio Amateurs Zoetermeer

Oktober 2020

Met in dit nummer:

- Onweerdetector V3
- Yaesu 817 uitbreiding
- Opa Vonk: Opstralingshoeken
- Hoe ik een WSPR zender bouwde - 2
- PA3CNO's Blog
- Afdelingsnieuws



Colofon

RAZzies is een uitgave van de Radio Amateurs Zoetermeer. Bijeenkomsten van de Radio Amateurs Zoetermeer vinden plaats op elke tweede en vierde woensdag van de maanden september - juni om 20:00 uur in het clubhuis van de Midgetgolfclub Zoetermeer in het Vernède sportpark in Zoetermeer.

Website:

http://www.pi4raz.nl

Redactie:

Frank Waarsenburg
PA3CNO
pa3cno@pi4raz.nl

Eindredactie:

Robert de Kok
PA2RDK
pa2rdk@pi4raz.nl

Informatie:

info@pi4raz.nl

Kopij en op- of aanmerkingen kunnen verstuurd worden naar razzies@pi4raz.nl

Nieuwsbrief:

http://pi4raz.nl/maillist/subscribe.php

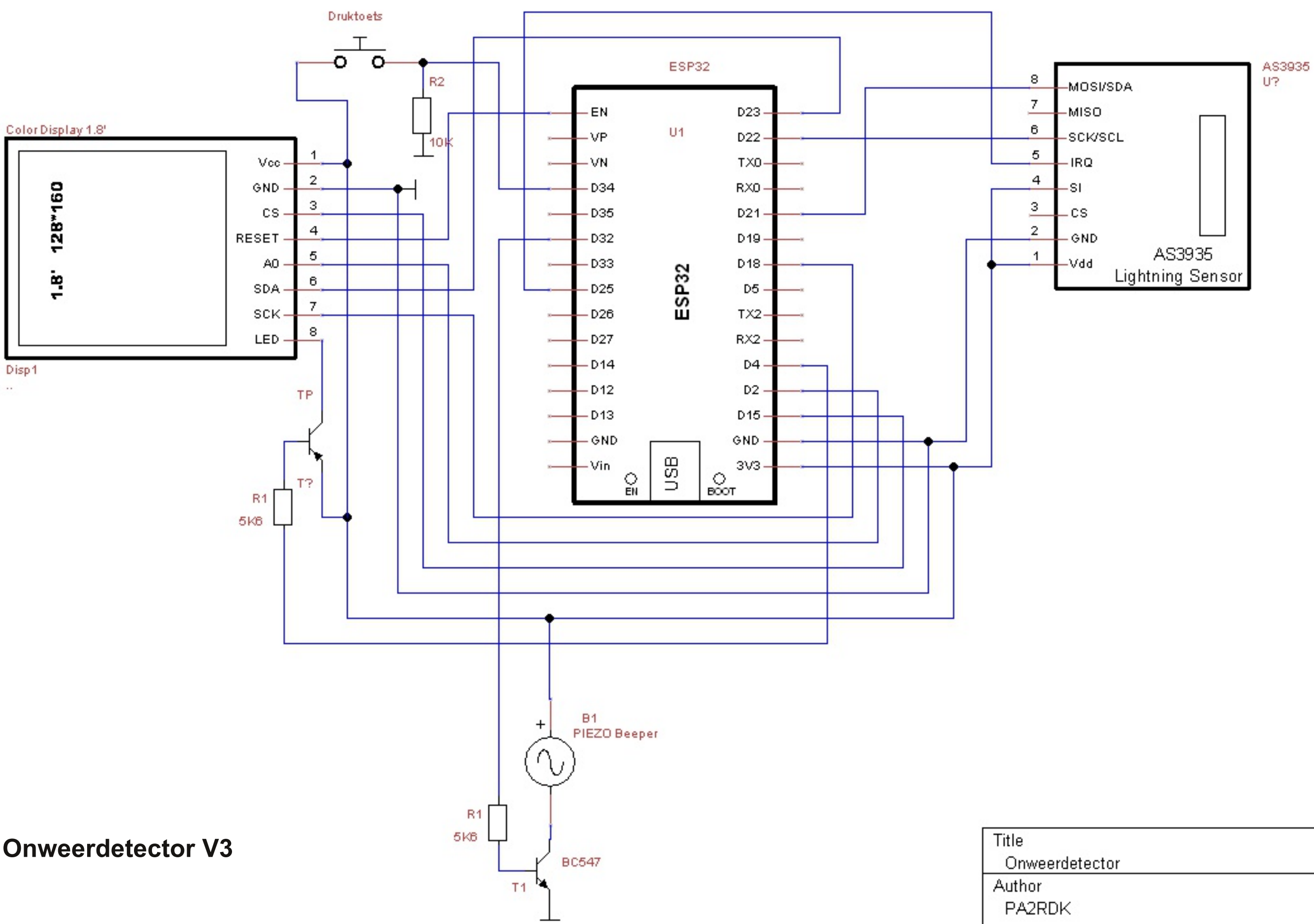
Van de redactie

Ik moet me even verontschuldigen voor de wat rommelige indeling van de uitgave van deze maand. Na enig experimenteren in de beginfase van ons maandblad ben ik uitgekomen op een indeling van twee kolommen. Dat leest voor een online magazine het prettigst, met uitvullen over de hele kolom breedte. Als je de eerste RAZzies leest zal je zien dat de indeling en opmaak nog wel een wisselde. Artikelen worden vaak aangeleverd in Microsoft Word, en daarbij wordt een heel A4-tje gebruikt om tekst weer te geven. Dat laat zich doorgaans door mij wel in twee kolommen gieten. Maar het wordt anders als er illustraties bij worden gevoegd die de hele paginabreedte vullen. Want dat in een kolom persen komt de details meestal niet ten goede, en een plaatje over een hele breedte plaatsen geeft het dilemma met de tekst: ga je onder het plaatje door, of vul je de kolommen boven het plaatje en ga je eronder verder? Als lezer moet je niet hoeven zoeken waar de tekst ergens verder gaat: dat moet intuïtief zijn. Ik kwam er deze maand niet uit. Vandaar dat een van de artikelen na de introductie in twee kolommen, overgaat in een pagina-breed verhaal. En aangezien de rest van het blad op de standaard wijze met de twee kolommen gevuld is, geeft dat in mijn ogen een rommelige indruk. Het is deze maand even niet anders, en ik hoop dat jullie deze manier van opmaken niet al te storend vinden.

Onweerdetector V3

Ik schreef er al over in mijn blog vorige maand: het overlijden van mijn onweerdetector als gevolg van eigenwijsheid van de eigenaar. Te weten: niet de modificatie doorvoeren om de WiFi module op 3,3V te zetten maar op 5V te laten staan. Het was lang goed gegaan, maar uiteindelijk gaf hij toch de geest. Ik had wel nieuwe WiFi modules besteld dus een dezer dagen zal ik 'm wel repareren, maar intussen wilde ik toch eens de nieuwe versie proberen die ontwikkeld was door Robert PA2RDK en door hem en Gert PE0MGB gebouwd werd. Met wisselend succes, want die van Gert gaf geen krimp terwijl de bliksem links en rechts ons dorp verlichtte. Dus heb ook ik deze versie nagebouwd.

Eigenlijk is deze versie niet zo heel veel verschillend van de vorige: het belangrijkste verschil is dat er geen separate WiFi module meer gebruikt wordt. Door gebruik te maken van een ESP32 als processor in plaats van de Arduino Nano, is de WiFi module al geïntegreerd met de ESP processor. En het goede nieuws is dat de ESP32 te programmeren is met de Arduino Software Development Kit (SDK). Een ander verschil is het display: een 128x128 schermpje biedt voldoende mogelijkheden voor het weergeven van informatie en grafieken. Laat je niet op het verkeerde been zetten door



Onweerdetector V3

Title Onweerdetector		
Author PA2RDK		
File ert\Documenten\TinyCad\Onweerdetector_ESP32		Document
Revision 3.1	Date 30-6-2020	Sheets 1 of 1

het schema, waar in het display 128x160 vermeld is. Daar trapte ik ook in, waarna de aansluitingen niet klopte en ik er pas na discussie met Robert achter kwam dat ik een 128x128 had moeten hebben. Let daar dus op.

Wat onder de motorkap anders is, is de zogenaamde Library (in het Nederlands Biblio-theek, maar programmeurs houden doorgaans de Engelse term aan). In de Nano software werd de PWFusion_AS3935 library gebruikt, maar die werkt niet met de ESP32. Zoals ik in mijn blog schreef, wordt in deze library gebruik gemaakt van de AVR architectuur en die is er niet in de ESP32. gelukkig is er voor de ESP32 ook een AS3935 library: de SparkFun_AS3935 library. Wat kan er mis gaan zou je zeggen. Tenslotte gebruikte ik dezelfde AS3935 Lightning Detector module in dit ontwerp, dus alleen wat ik tegen de module zeg in de software maakt uit. En als ik hetzelfde zeg, moet hij hetzelfde reageren was mijn redenering.

Maar zo was het niet helemaal. De module leek ongevoelig, en áls hij al reageerde, gaf hij de afstand tot de ontlading steevast als 1 km weer. En hoe ik ook rommelde met de software, hij reageerde gewoon niet zo goed als mijn oude detector met losse WiFi module.

Nou meende ik me te herinneren dat Robert in de oude versie een hele tijd bezig was geweest om die afstanden goed te krijgen. In de datasheet van de AS3935 staat dan ook dat de afstand niet de afstand tot de ontlading is, maar de afstand tot het front van de onweersbui. Na elke gerapporteerde "strike" resette Robert de module door na het afhandelen van de interrupt routine een ClearDistance en een ClearStatistics te doen. Alleen bestond de ClearDistance routine wel in de PWFusion library, maar niet in de SparkFun library. Daarnaast was ook de timing van de interrupt afhandelingsroutine anders: op het kritieke randje. Ik paste dus de SparkFun library aan door de timing aan te passen en de ClearDistance routine toe te voegen. Door ook de ReadAllRegisters routine toe te voegen die kennelijk uitmaakt voor de timing van de module, begon het weer een beetje te werken zoals de oude module deed. Er verschenen afstanden van 1 tot zo'n 10km. Maar niet meer.

Nou gebruikte ik voor de voeding van de onweerdetector een stekkernetvoeding die zo'n 12V afgeeft. Die gebruikte ik ook met mijn oude detector, alleen zat daar nog een heel accu-laadcircuit tussen en de 7,2V van de accu's werd toegevoerd aan de Vin van de Arduino Nano. Die heeft een spanningsregelaar aan boord die uiteindelijk voor de 5V zorgt. Maar dat heeft de ESP32 niet. Ik gebruikte voor de V3 detector gewoon een 7805 die van de 12V stekkernetvoeding 5V maakt.

Tijdens een van de experimenten waarbij ik de software opnieuw programmeerde in de ESP32, had ik de voeding niet aangesloten maar voedde ik de ESP32 via de USB aansluiting van de computer. En toen gaf de detector wél de afstanden goed weer. Het lijkt er dus op dat via de voeding rotzooi binnenkomt die invloed heeft op de waarnemingen van de detectormodule. Dat had Gert ook al eens vastgesteld, maar ik had het zelf nog nooit gemerkt. Nu dus wel. 500kHz, de frequentie waarop de onweer-module ontvangt, is in een bebouwde omgeving nou niet direct de meest stille deel van het spectrum en een schakelende voeding helpt daar niet bij. Besteed dus aandacht aan de voeding als je de detector nabouwt.

Daarover gesproken: er is geen printje van. Zo ingewikkeld is het schema nou ook weer niet en een stukje veroboard met een paar draadjes doen wonderen. Zie b.v. hoe Gert zijn all-band radio maakte met gelakt draad in de RAZzies van september dit jaar. De linkjes naar de sketch en de verbouwde library vind je hieronder. Gebruik deze library, omdat die speciaal aangepast is voor de versie 3!

Link naar de sketch

Link naar de aangepaste library

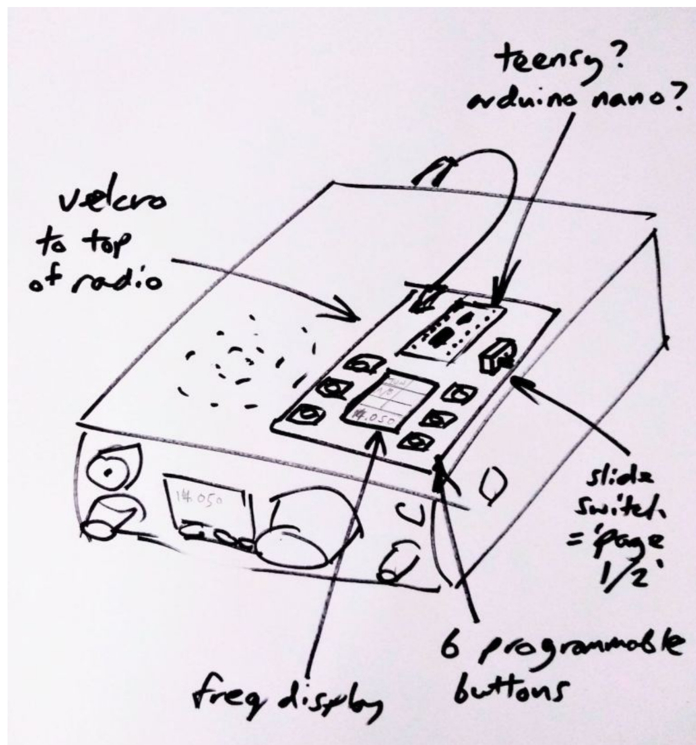
Uitbreiding voor Yaesu FT817

Andy Webster G7UHN



Zoals zovelen vind ik het heerlijk om portable te werken met mijn FT-817, maar ik besteed daarbij veel van mijn werktijd aan het spelen met de softkeys, omdat mijn meest gebruikte functies (CW-smal filter, vermogens- en keyer-instellingen om een ATU af te stemmen, A/B, A=B, etc.) verspreid zijn over verschillende “pagina's” van de A, B, C knoppen. Vergeleken met de sublieme ervaring bij het gebruik van mijn Elecraft K2, kan de FT-817 een beetje frustrerend zijn!

Vorige maand sloeg de inspiratie toe en ik bedacht dat ik wellicht een kleine microcontroller en een klein OLED-scherm met een paar knoppen in elkaar zou kunnen prutsen om wat extra softkeys voor de radio te definiëren via de CAT-seriële poort.



Niets bijzonders eigenlijk (ik heb artikelen gezien van mensen die PIC's voor dit doel gebruiken), maar het leek me een mooi groot project om mee te spelen en wat ervaring op te doen met het maken van printen (ik heb dit thuis maar één keer eerder gedaan). Een klein beetje discussie met Michael G0POT (FT-817 en SOTA-goeroe), wat Google-zoekopdrachten, het raadplegen van de uitstekende referentiepagina van KA7OEI
(http://www.ka7oei.com/ft817_meow.html)
en wat filosoferen over onze favoriete FT-817 instructies leverde voldoende inspiratie...

Toevallig had ik het geluk dat de juiste onderdelen (Arduino Nano, klein OLED-display, knoppen, prototypebord en een 8-pins mini DIN-kabel) in huis had rondslingerden om die avond het 'eerste licht' te zien uit de seriële poort van mijn FT-817. De Arduino Nano is een goede comtroller om mee te beginnen omdat hij op 5V werkt en dus rechtstreeks kan werken met de spanningsniveaus op de ACC-poort van de FT817. Wat daarna volgde, waren een paar nachten van hacken van de Arduino-code om de data voor mijn favoriete commando's te kunnen verzenden en ontvangen en nog wat meer experimenten op het prototype-bord.



Ik heb een paar goedkope OLED-schermen geprobeerd en ze zien er binnenshuis geweldig

uit, maar waren niet goed genoeg in de volle zon, waarin ik meestal portable werk (zie foto op de vorige bladzijde)



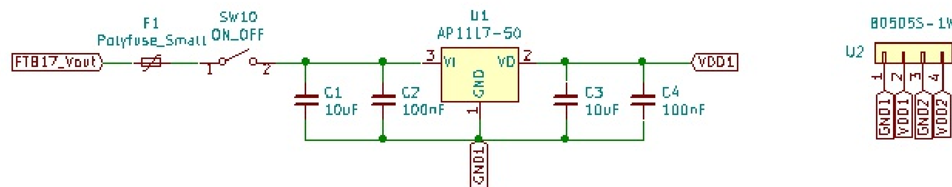
Op dit punt had ik me ook gerealiseerd dat het wel makkelijk zou zijn om een extra display bovenop de radio te hebben dat veel gemakkelijker af te lezen is dan het eigen display van de 817 op het voorpaneel. Ik had tijdens het testen wat storing ondervonden van het niet-afgeschermde prototypebord dat te horen was als klikgeluiden in de ontvangst, dus het leek erop dat enige afscherming tussen de radio en mijn schakeling noodzakelijk was. Na veel internet zelfstudies besloot ik over te schakelen op het gebruik van een LCD-scherm in Nokia 5110-stijl voor betere leesbaarheid bij daglicht en een lager energieverbruik. Het toevoegen van een ADUM1201 digitale isolator en een B0505S-1W geïsoleerde DC-DC-omzetter aan het prototype-bord (modules die zeer snel door eBay-leveranciers geleverd werden) gaf me wat isolatie en verlaagde het stoorniveau waarvan ik vermoedde dat die uiteindelijk helemaal zou verdwijnen als ik het ontwerp op print zou zetten met goede aardvlakken rond de signaallijnen.

Met een (grotendeels) werkend prototype was het tijd om de tutorials op internet opnieuw te bestuderen, deze keer om te leren hoe je KiCad, een gratis open-source PCB-ontwerptool die beschikbaar is op Linux, Windows en Mac, moet gebruiken. Ik heb ooit één keer eerder een print gemaakt voor een thuisproject met gebruik van Autodesk EAGLE en ik vond het behoorlijk moeilijk om Eagle te leren; het lijkt erop dat het 20 jaar aan bagage en dogma's in de gebruikersinterface met zich meedraagt. In feite begon ik in eerste instantie EAGLE te gebruiken voor dit project, maar ik heb de eerste avond 3 uur besteed aan het proberen om de labels op de ADUM1201-chip te veranderen die ik niet kon vinden in een EAGLE-bibliotheek... dus ik gaf het op en besloot ik om KiCad te proberen, dat ik recentelijk in een paar goede recenties was tegengekomen. Ik ben blij te kunnen melden dat ik na het vinden van een uitstekende tutorial over KiCad, in ongeveer 15 uur werktijd verspreid over een paar avonden het schema en mijn print-layout had getekend.

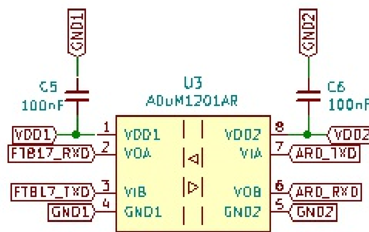
Ik moet hieraan toevoegen dat in de 15 uur KiCad-tijd een aantal uren verbrand werden aan de keuze van de schuifschakelaar, zodat een print veel sneller kan worden ontworpen als je eenmaal je favoriete onderdelen hebt gesorteerd!

Dat is behoorlijk indrukwekkend voor mijn eerste poging met KiCad als een bijna-beginner met het maken van printen, ik kan het van harte aanbevelen, KiCad was zoveel gemakkelijker dan EAGLE en best een plezierige tool. Goed, print-ontwerp gereed en geüpload naar JLCPCB voor fabricage. 5 printen met DHL-verzending kostten me minder dan £20 en kwamen binnen 5 kalenderdagen uit China aan. Er zijn trouwens diverse print-fabrikanten...

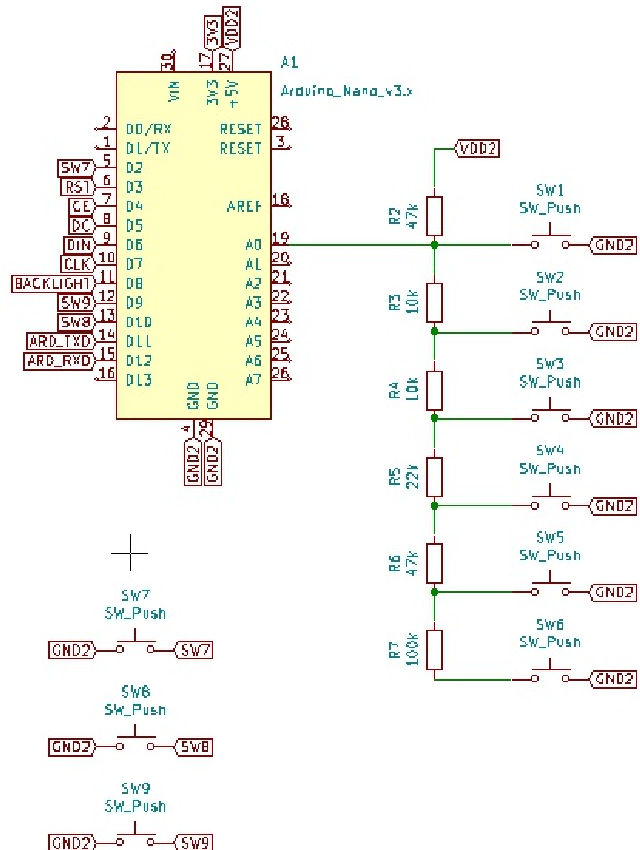
Dus dat brengt ons zo ongeveer bij vandaag. De printplaat was zeer snel (!) in elkaar gezet en er is geen teken meer van storing van de seriële datalijnen die de ontvanger van de 817



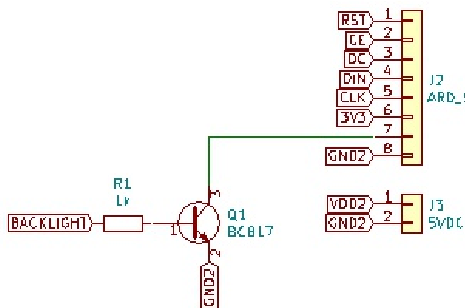
F1
Polyfuse_Small
SW10
ON_OFF
FT817_Vout
U1
AP1117-50
C1
10uF
C2
100nF
C3
10uF
C4
100nF
VDD1
GND1



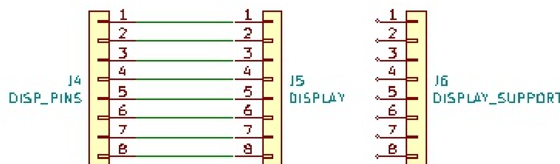
U3
ADuM1201AR
C5
100nF
C6
100nF
VDD1
FT817_RXD
FT817_TXD
GND1
VDD2
ARD_TXD
ARD_RXD
GND2



A1
Arduino_Nano_v3.x
SW7
RST
CE
DC
DIN
CLK
BACKLIGHT
SW9
SW8
ARD_TXD
ARD_RXD
3V3
VDD2
GND2
R2
47k
R3
10k
R4
10k
R5
22k
R6
47k
R7
100k
SW1
SW_Push
SW2
SW_Push
SW3
SW_Push
SW4
SW_Push
SW5
SW_Push
SW6
SW_Push



J2
ARD_SPI
RST
CE
DC
DIN
CLK
3V3
GND2
J3
5VDC
VDD2
GND2
R1
1k
Q1
BC817
BACKLIGHT
GND2

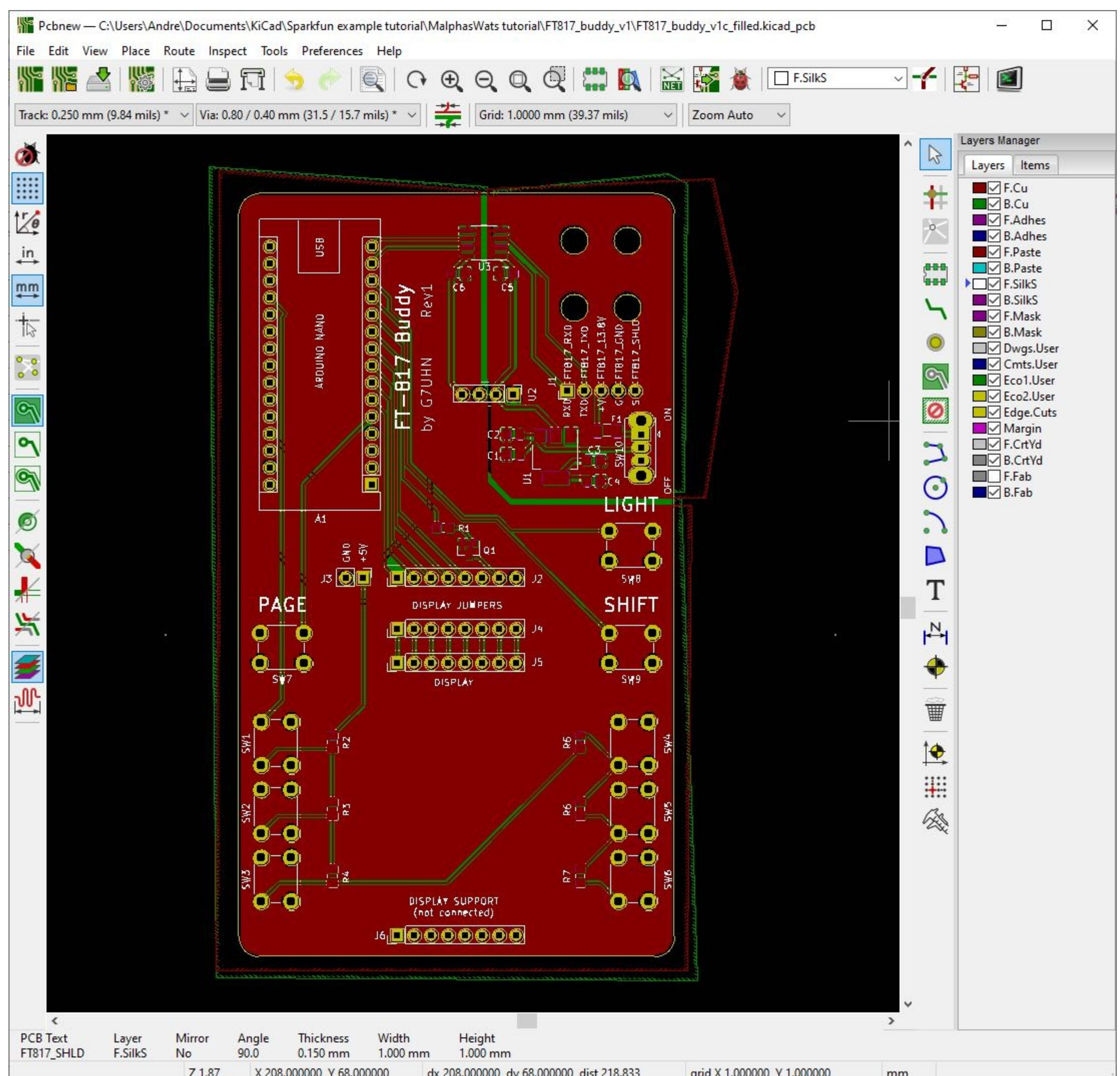


J4
DISP_PINS
J5
DISPLAY
J6
DISPLAY_SUPPORT

binnensluipen, nu de onderdelen op de print gemonteerd zijn. Sommige lessen zijn geleerd tijdens de bouw (bijv. bruine 6 mm drukknoppen zijn minder "klikkerig" dan de zwarte en dat is maar goed ook!) en ik heb nu mijn FT-817 display/knoppen uitbreiding in veldtest. Ik ben niet van plan om dit te verkopen, het is een vrij eenvoudig ontwerp, maar het is een geweldig huisproject om je vaardigheden op het gebied van microcontrollers, printen bestücken en bouwen te verbeteren. Ik zal te zijner tijd een artikel op [mijn website]() plaatsen.

Tijdens gebruik werkt het apparaat precies zoals ik had gehoopt, ik kan alles doen wat ik wil op mijn FT-817 zonder dat ik door de ongemakkelijke reeks drukken op knoppen moet worstelen. Het frequentiedisplay bevindt zich nu ook in een veel betere positie voor mij (zoals de meeste FT-817-eigenaren zullen weten, aangezien ze met een scheef oog naar de KX2, KX3, enz. kijken...!) En volgens mij heb ik uitsluitend de uitbreiding gebruikt in het veld op zaterdag. Was mijn CW maar zo goed geweest!

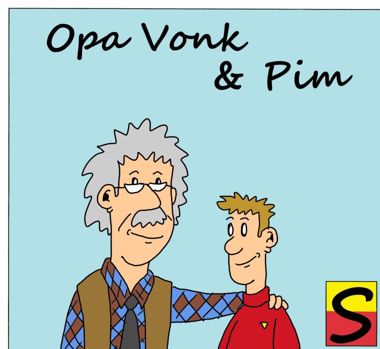
De volgende stappen zijn om aan de Arduino-code te werken. Mijn code is behoorlijk rommelig (mijn coderingsstijl omvat veel Stack Overflows en kopiëren/plakken!) en is niet geschikt voor algemeen gebruik. Er zijn ook enkele waarschuwingen die in acht moeten worden genomen bij het manipuleren van de



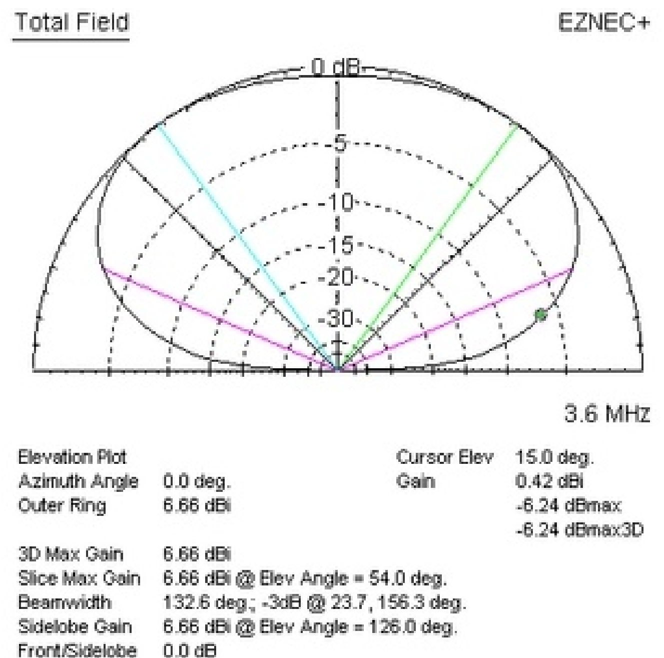
EEPROM van de FT-817 (vereist voor sommige van de functies die ik wilde), die zijn uitgelegd op de KA7OEI-pagina, maar er zijn gelukkig een paar vrijwilligers op Twitter geweest om me te helpen met de software, wat geweldig is. Ook wil ik nog een "Rev 2" -bord maken met een Arduino Pro Mini om de belasting van de FT-817-accu te verlagen voordat de print-bestanden worden gedeeld. Behalve dat is het nu tijd om weer naar buiten te gaan en te genieten van de nieuwe verbeterde interface voor mijn kleinste radio!

73 Andy G7UHN

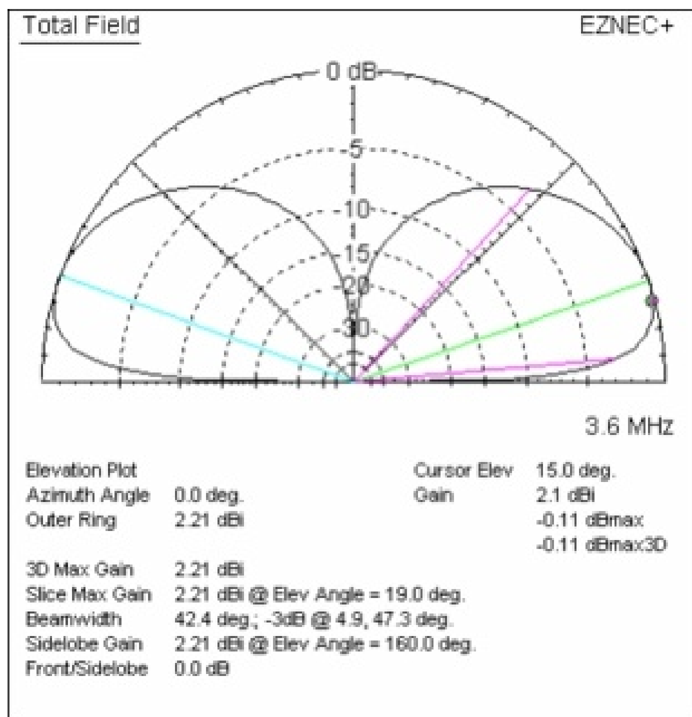




Pim hing half uit het raam van zijn Opa Vonk's shack in een poging de antennes voor HF goed te kunnen zien. "Weet je dat de ziekenhuizen momenteel vol liggen met Corona patiënten en dat ze niet op jou zitten te wachten? Wat ben je in vredesnaam aan het doen?" vroeg Opa. Pim liet zich weer naar binnen vallen en zei: "Ik probeer een inschatting te maken van de hoogte en plaatsing van de antennes, zodat ik weet wat ongeveer de opstralingshoek zal zijn van de antennes. Dan kan ik een betere keuze maken als ik DX of dichtbij wil luisteren", zei Pim. Opa fronste een wenkbrauw, een teken voor Pim dat hij iets gezegd had waar Opa een mening over had. En dat Opa die mening ook wel zou laten weten. "Laat me raden", zei Opa. "Jij hebt ergens gelezen dat een lage opstralingshoek goed is voor DX werk, en dat een hoge opstralingshoek goed is voor lokaal verkeer of de wat kortere afstanden". Pim knikte. "De opstralingshoek is een zwaar overschat gegeven", zei Opa. "Ik zal je een paar voorbeelden geven waaruit blijkt dat de opstralingshoek lang niet zo bepalend is als jij denkt. Kijken we eerst maar eens naar een vertical in vergelijking met een dipool. Iedereen weet dat een vertical een lage opstralingshoek heeft en dat een dipool op een niet al te grote hoogte beter is voor de wat kortere afstanden vanwege zijn hoge opstralingshoek. Laten we eens kijken wat daar van waar is.



Deze dipool is opgesteld op een hoogte van 1/3 golflengte. De maximale versterking is 6.66 dBi en hij heeft een opstralingshoek van 54 graden (de lichtblauwe lijnen in de grafiek). Onder een hoek van 15 graden (het grijze balletje rechts in het stralingsdiagram) is de versterking van de dipool nog 0.42 dBi. Er zijn maar weinig amateurs die, als ze uitsluitend naar de opstralingshoek van deze dipool kijken, deze antenne zouden kwalificeren als 'geschikt voor DX', of zelfs maar voor mogelijk zouden houden dat je er überhaupt DX mee zou kunnen werken.



Laten we nu eens kijken naar een vertical met 16 radialen, zie de grafiek hierboven. Deze antenne heeft zijn maximale versterking bij een hoek van 19 graden (lichtblauwe lijnen), welke dan 2.21 dBi is. De versterking bij 15 graden, de hoek die als zeer geschikt beschouwd wordt voor verre DX op de lage HF banden, is 2.1 dBi (weer het grijze balletje rechts op de grafiek). Deze vertical heeft slechts 1.7 dB meer versterking dan de zogenaamde waardeloze dipool bij de zeer gewenste lage-band DX opstralingshoek van 15 graden! Bij 19 graden, waar de maximale versterking van de vertical optreedt, hebben de dipool en de vertical feitelijk dezelfde versterking, ook al heeft de vertical een opstralingshoek van 19 graden en de dipool een opstralingshoek van 54 graden.

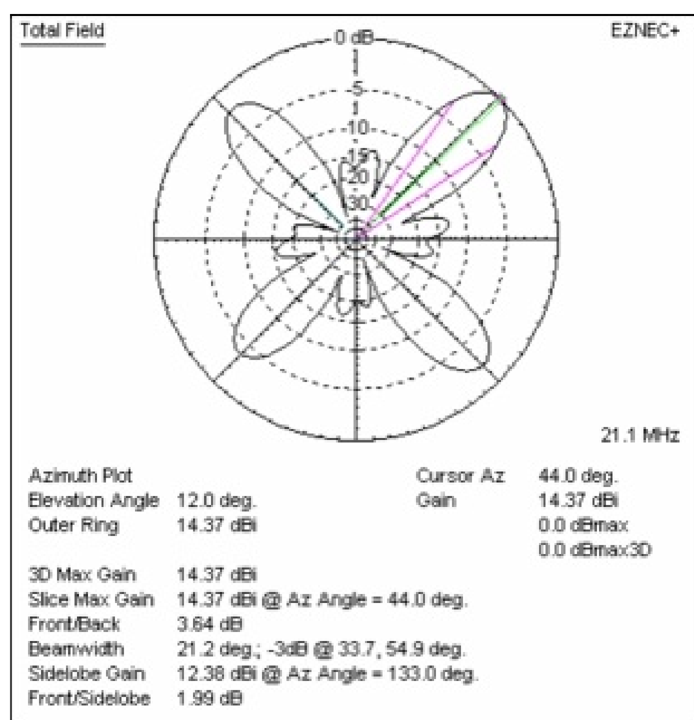
Dat betekent niet dat de vertical onbruikbaar is voor verbindingen met dichtbij gelegen stations, maar hij is wel zwaar in het nadeel ten opzichte van de dipool. Het betekent ook niet dat de dipool niet bruikbaar is voor DX, of niet soms sterker zal zijn dan de vertical over zeer grote afstanden. Kleine verschillen in de locatie en installatie kunnen de dipool zomaar de betere antenne maken voor grote afstanden. Het enige dat wel zeker is, is dat de vertical niet erg geschikt zal zijn voor grote opstralingshoeken.

Het moge duidelijk zijn dat de opstralingshoek op zich maar weinig zegt over hoe goed een antenne geschikt is voor DX. En dan wil je natuurlijk weten wat dan wél belangrijk is. Waar moet je dan op letten?

Het antwoord is pijnlijk eenvoudig: we willen de maximale beschikbare veldsterkte onder alle hoeken en in alle richtingen. Maxima en minima buiten de gewenste richtingen en hoeken laten we buiten beschouwing waar het gaat om het bereiken van het beste signaal in de door jou gewenste richting. Je moet je dus helemaal niet druk maken over de opstralingshoek: het enige wat je wil is de maximale veldsterkte in de gewenste richting!

Dat brengt me op een zwaar onderschat probleem: stralingsminima. Ik noemde het al even. We willen dus een maximaal signaal in de gunstigste richting onder de gunstigste hoek, maar dat moeten we nooit verwarren met de maximale versterking in die richting. De maximale versterking op zich is net zo'n nutteloos getal als de opstralingshoek. Laten we eens naar een paar voorbeelden van hoge maar nutteloze versterking kijken.

Stel je voor dat je midden in Europa woont en dat je het grootste deel van Europa wil kunnen

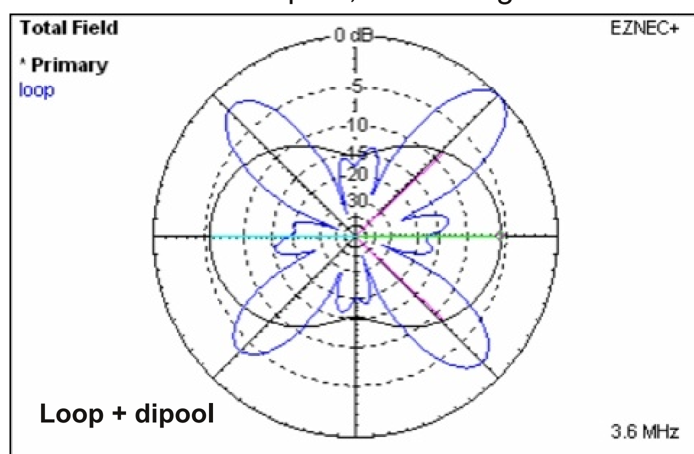


werken met je multiband antenne. Wellicht is het je bekend dat een loop voor de lage banden die je gebruikt in het hoge deel van de HF band, een behoorlijke versterking heeft. Dat is gunstig, nietwaar? Tenslotte wil iedereen een antenne met veel versterking.

In de grafiek onder aan de vorige bladzijde zie je het stralingsdiagram van een hele-golf loop antenne voor 80m, gebruikt op 21MHz. Die 80m loop heeft bij 21MHz een versterking van 14.37 dBi, ongeveer 6 dB meer dan een dipool. Hoewel deze versterking erg aantrekkelijk klinkt, is deze extra 6 dBd veldsterkte verdeeld over vier grote en vier kleine lobben. De gemiddelde versterking in de grote lobben is 12.46dBi en dat is ongeveer 4 dB meer dan een dipool. Maar wat erger is, is dat die versterking slechts geldt in een openingshoek van 21 graden voor elk van de lobben, en naast die lobben zitten minima tot 30 dB diep!

Dus hoewel je nu kunt opscheppen over de hoge versterking ten opzichte van een isotrope straler (het dBi getal), treedt die versterking maar op in een zeer smal gebied. In het gunstigste geval is de versterking gemiddeld 4 dB hoger dan een dipool in vier zeer specifieke richtingen van 44, 133, 224 en 315 graden. De -4db punten van de hoofdlobben, waar de versterking precies gelijk is aan de maximale versterking van de dipool, liggen op een bundelbreedte van ongeveer 11 graden.

Statistisch gezien zal het erg moeilijk zijn om binnen die hoofdlobben, waar het signaal harder is dan dat van de dipool, verbindingen te maken!



Loop + dipool

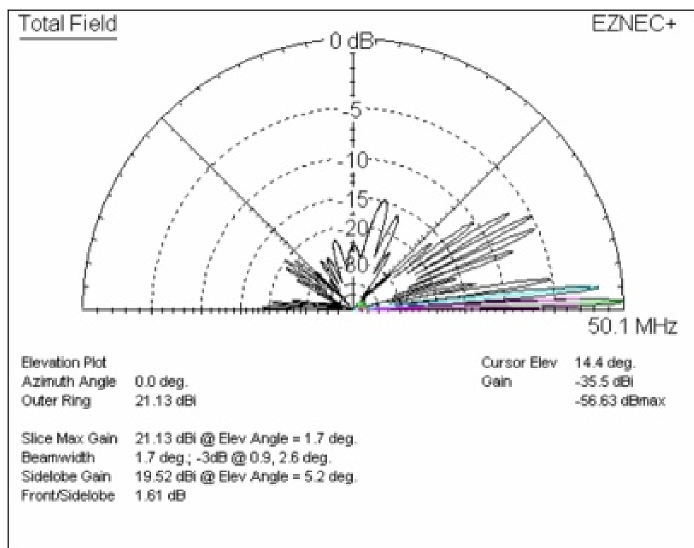
Een vergelijking met een simpele dipool antenne (plaatje linksonder) toont aan dat de loop 8dB beter is in vier zeer specifieke richtingen met kleine bundelbreedte. Het horizontale vlak waarin de loop gelijk of beter presteert dan de dipool is 125 graden. Dat wil dus zeggen dat de dipool in een vlak van 235 graden beter presteert dan de loop! Dat betekent dat je vanuit bijna elk deel van Europa betere signalen ontvangt met een kleine halve-golf dipool antenne dan met de loop.

Dit leidt tot een regel die we vaak verwaarlozen of vergeten. Als je de minima in het stralingspatroon niet kunt verplaatsen, bijvoorbeeld door een draaibare antenne, dan kan je beter een regelmatig stralingsdiagram hebben met een beetje minder versterking. Het laatste wat je wil is meerdere minima in bruikbare richtingen.

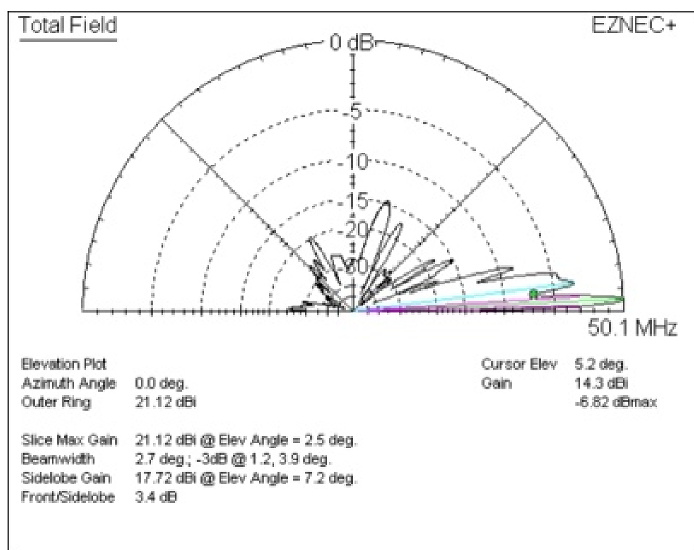
Wat geldt voor het horizontale stralingspatroon, geldt ook voor het vertikale patroon. Alweer: het laatste wat je wil is meerder diepe minima verspreid over bruikbare opstralingshoeken. Een wat breder elevatiepatroon (zonder dat dit effect heeft op de efficiency of de dekking in het horizontale vlak) resulteert in iets minder versterking, waarbij je 3dB verliest als de elevatiebundel twee keer zo groot wordt. Dat gaat nooit het verschil maken tussen iemand die ons S9 hoort met de ene antenne en helemaal niet met de andere antenne. Op dezelfde manier gaat een opstralingshoek die van 30 graden naar 5 graden gaat, dat verschil ook niet maken.

De enige uitzondering hierop is als we een patroon hebben met een diep minimum dat toevallig precies in de doelhoek of richting valt. De minima in het vertikale stralingspatroon veroorzaken de problemen van aanzienlijk signaalverlies, niet de versterking of de opstralingshoek!

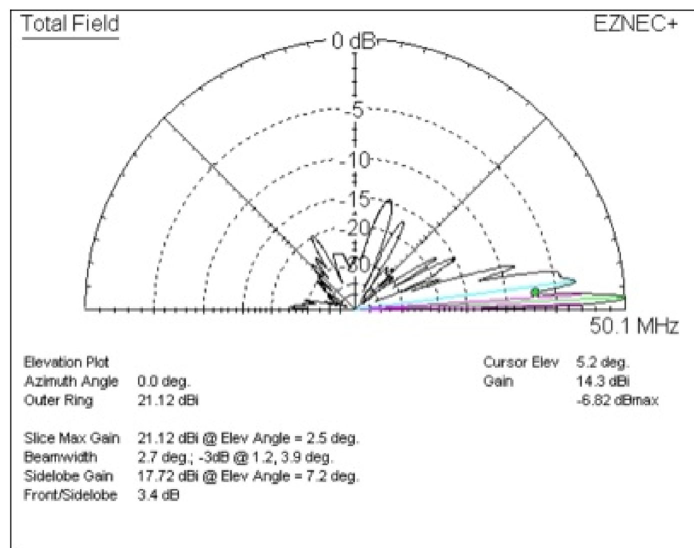
Laten we eens kijken naar het patroon van een stack van 6m antennes die verbonden zijn met traditionele faselijnen (dat zijn coaxen met een bepaalde lengte die ervoor zorgen dat het signaal op meerdere antennes in fase is).



Met dit patroon zou de propagatie erg grillig lijken. Een verandering in invalshoek van slechts 2 graden naar 4 graden zou kunnen resulteren in een signaalvermindering van 20 dB. Dit is geen opstralingshoek-probleem. Het probleem zit 'm in smalle hoofdlobben met diepe minima tussen scherpe pieken. Als de invalshoek van het signaal slechts 1-2 graden varieert, zou het signaalniveau van uitstekende leesbaarheid naar onleesbaar kunnen gaan.



Een methode om de diepe minima te corrigeren, is door antennes toe te voegen en een progressieve faseverschuiving tussen antennes te gebruiken. Aangezien antennes normaal in de richting van de achterblijvende fase stralen, en omdat we de minima boven de hoofdlob willen vullen, zou de logische faseverschuiving een progressieve achterblijvende faseverschuiving zijn met steeds hogere antennes. Dat wil zeggen dat de hogere antennes in de stack achterblijven in de fase ten opzichte van de antennes daaronder.



In bovenstaande figuur resulteert de toevoeging van nog twee antennes met een achterblijvende progressieve verschuiving van 30 graden in een array die veel minder gevoelig is voor de invalshoek van het signaal. Dit geeft vrijwel een constante veldsterkte, binnen 6 dB (1 S-punt), van 0,9 graden tot 11 graden. Hij heeft nog steeds bijna dezelfde piekversterking bij zeer lage hoeken om de band te openen, of om te gebruiken wanneer de MUF net boven de werkfrequentie ligt, maar heeft niet het probleem met 20-30 dB diepe minima als de invalshoek slechts 2 tot 4 graden vanaf een lobpiek verschuift. Een dergelijk patroon zou in de loop van de tijd veel gelijkmatiger signalen opleveren.

Kortom. Je kunt wel uit het raam gaan hangen om naar de antennes te kijken, maar je kunt beter naar het stralingspatroon kijken in de richting waarin je verbindingen wil maken. En ja, een vertical heeft een lage opstralingshoek. Maar dat maakt 'm niet per definitie beter dan een dipool. Integendeel: zoals ik je liet zien, doet de versterking van een dipool in de richting waar de vertical zijn maximale opstraling heeft, helemaal niet zoveel onder ten opzichte van de versterking van de vertical. Dus als ik jou was, zou ik gewoon de dipool gebruiken", besloot Opa zijn verhaal. Pim knikte dat hij het begrepen had. "Ik had echt gedacht dat het veel verschil zou maken", zei hij. "Maar ik ben weer wat wijzer geworden. Ik ga de dipool gebruiken".

Hoe ik een WSPR zender bouwde – deel 2: codering van de informatie

Ruud Jongeling PE2BS

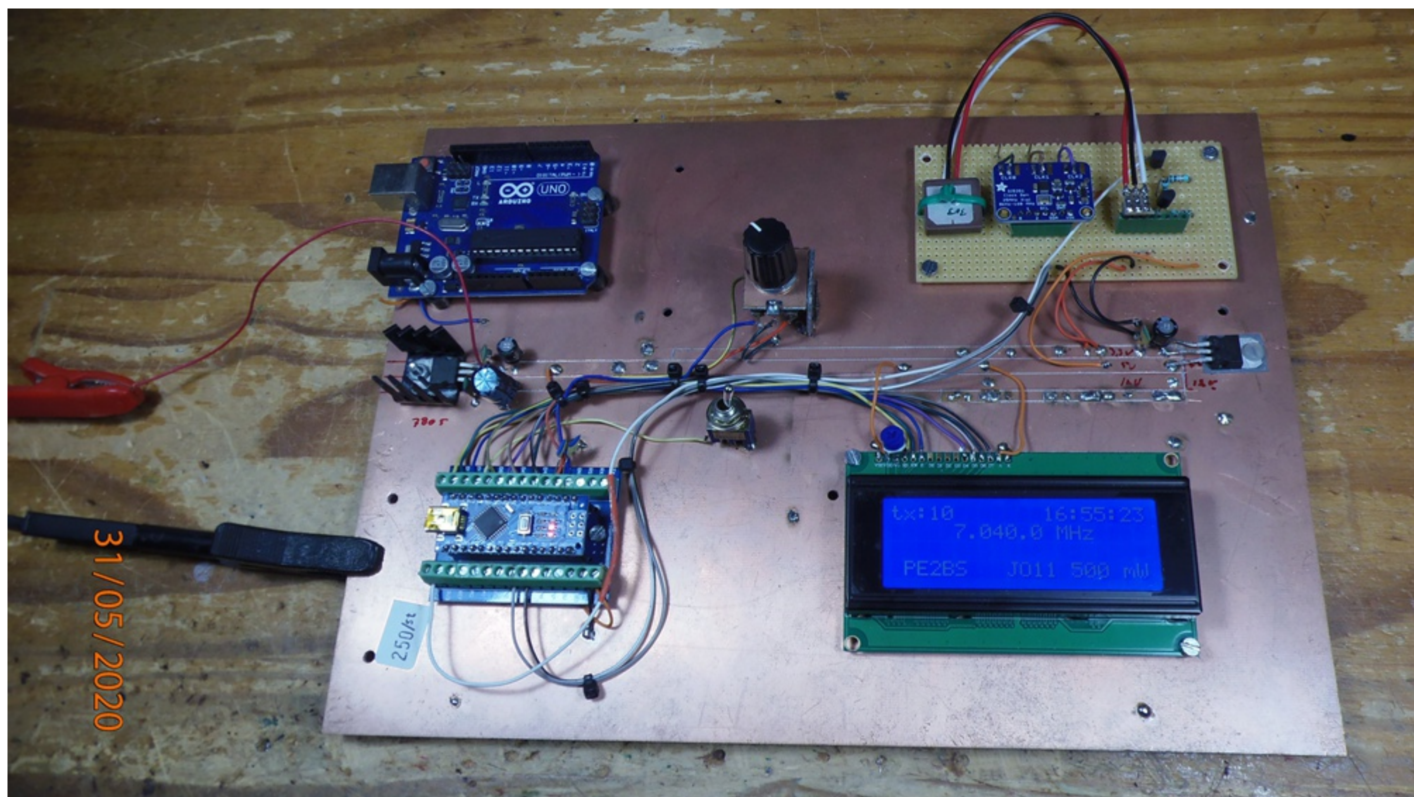
In het eerste deel van deze serie hebben we gekeken naar de hardware van de zender. In dit tweede deel gaan we in op wat ik zelf het lastigste onderdeel van het project vond: het omzetten van de call, locator en vermogen naar een array van 162 bytes met waarden tussen de 0 en 3. Om de WSPR zender te kunnen aansturen worden de call van het station, de locator en het vermogen in verschillende stappen gecodeerd en vertaald naar een 4-toon FSK signaal. De handleiding van WSPR 3.0 geeft in grote lijnen aan hoe een ander wordt aangepakt maar was voor mij te beknopt om van daaruit een programma voor de Arduino Nano te schrijven. Een betere toelichting heb ik gevonden in het artikel van Andy Talbot (G4JNT) *The WSPR Coding Process* maar ook dit artikel liet op onderdelen vragen open. Het is uiteindelijk wel gelukt maar het was best even puzzelen. Daar moet je natuurlijk wel van houden. Je kunt me ook een mail sturen (pe2bs@kpnmail.nl) dan krijg je het ino-bestand toegestuurd.

De codering van de call, locator en vermogen verloopt in 5 stappen:

1. Omzetten van de call, locator en vermogen in twee unsigned long variabelen.
2. Deze twee variabelen omzetten naar een array van 11 bytes.
3. Deze 11 bytes voorzien van een voorwaartse fout correctie (FEC) waardoor een array van 162 bits ontstaat.
4. De volgorde van de 162 bits uit de derde stap aanpassen.
5. De array van 162 bits uit de vierde stap combineren met een synchronisatie vector tot een array van 162 bytes met een waarde tussen de 0 en 3, de 4 FSK tonen.

<u>Eerste stap: coderen van de call, locator en vermogen</u>

De call, de locator en het vermogen worden in twee getallen omgezet en bewaard in twee unsigned long variabelen. De call mag maximaal 6 symbolen lang zijn waarbij het derde symbool altijd een cijfer is. De lege plaatsen in zowel de



Afbeelding 2a: de ontwikkelomgeving van het programma

prefix als de suffix worden opgevuld met een spatie. Mijn eigen call (PE2BS) wordt dus PE2BS[SPATIE] en het voorbeeld uit de handleiding van WSPR (K1ABC) wordt [SPATIE]K1ABC. Voor de cijfers 0….9 wordt de waarden 0 t/m 9 gebruikt, voor de letters A..Z en SPATIE de waarden 10 t/m 36. De SPATIE heeft dus de waarde 36.
K1ABC wordt dan: 36 – 20 – 1 – 10 – 11 – 12. De kunst is nu de call (en alle varianten op deze call) in een zo'n klein mogelijk getal te verpakken. Ik had de call in een array call[6] van 6 bytes geplaatst. De unsigned long variabele N gaat nu de gecodeerde call bevatten:

```
// codering call
  N = call[0];
  N = N * 36 + call[1];
  N = N * 10 + call[2];
  N = N * 27 + (call[3]-10);
  N = N * 27 + (call[4]-10);
  N = N * 27 + (call[5]-10);
```

Listing 2a: coderen van de call in de variabele N

Voor de voorbeeld call K1ABC wordt dit:

N = 36	*de SPATIE] die in dit geval aan de call vooraf gaat*
N = 36 * 36 + 20	*de letter K wordt aan de spatie toegevoegd*
N = 1316 * 10 + 1	*het cijfer 1 wordt aan de call toegevoegd*
N = 13161 * 27 + (10-10)	*de letter A wordt aan de call toegevoegd*
N = 355347 * 27 + (11-10)	*de letter B wordt aan de call toegevoegd*
N = 9594370 * 27 + (12-10)	*de letter C wordt aan de call toegevoegd*
N = 259047992 = 0x0F70C238	

Voor de liefhebbers:

Hoe kun je uit het getal dat de call bevat weer terugkomen bij de call? Laten we het voorbeeld nemen van de prefix PE2.
Als getal wordt dit: ((P * 36)+E)*10+2 of wel ((25 * 36)+14)*10+2 = 9142

Terugrekenen gaat als volgt:
We hebben de P en de E als laatste vermenigvuldigd met 10 en daarna het cijfer uit de prefix erbij opgesteld: 9142 modulo 10 = 2 (modulo 10 is delen door 10 en je noteert de rest). Hier komt de 2 uit de call tevoorschijn.
De 2 halen we van het getal af: 9142 – 2 = 9140. Hierin zitten nu nog de letter P en E en dit geheel is met 10 vermenigvuldigd. We delen door 10 en houden 914 over waarin nog steeds de letters P en E zitten. De eerste letter van de prefix was vermenigvuldigd met 36. Dus we delen door 36 en noteren de rest: 914 modulo 36 = 14. Dit getal hoort bij de letter E.
De letter E kan nu uit het getal gehaald worden: 914 – 14 = 900. In dit getal zit nu nog alleen de eerste letter van de call vermenigvuldigd met 36. We doen 900 : 36 = 25 en daar is ook de letter P weer terug.

De locator zit in een array van 4 bytes locator[4] waarbij elementen 0 en 1 de letters van A t/m R kunnen bevatten en de elementen 2 en 3 de cijfers van 0 t/m 9. Ook de locator wordt in een zo klein mogelijk getal verpakt en geplaatst in een variabele M van het type unsigned long.

```
// codering locator
  M = (179 - 10 * (locator[0]-10) - locator[2])*180 + (10 * (locator[1]-10) + locator[3]);
```

Listing 2b: coderen van de locator in de variabele M

Let op de volgorde van de array elementen. Voor de locator uit het voorbeeld van de handleiding (FN42) bevat locator[0] in de array de letter F waarbij het getal 15 hoort. De locator[3] bevat de waarde 2 uit de locator. De berekening van locator FN42 wordt dan:

M = (179 -10 * (15-10) – 4)*180 + (10 * (23-10) + 2)
M = 22632 = 0x5868

Aan de variabele M met de locator wordt het vermogen van de zender toegevoegd. Het vermogen mag ieder getal zijn tussen de 0 en de 60. Bij de decodering van het WSPR signaal doen overigens alleen de getallen die eindigen op 0, 3 en 7 mee. De overige getallen mogen wel in de code zitten maar worden naar het dichtst bijzijnde vermogen afgerond.

```
// codering power
  M = M * 128 + pwr + 64;
```

Listing 2c: coderen van het vermogen in de variabele M

In het voorbeeld is voor het vermogen het getal 37dBm gekozen wat afgerond gelijk staat aan 5W. De berekening die hierbij hoort wordt:

M = 22632 * 128 + 37 + 64
M = 2896997 = 0x2C3465

De eerste stap van de vijf stappen is hiermee afgerond. De call, locator en het vermogen zijn als getal verpakt in de variabelen N en M van het type unsigned long.

<u>Tweede stap: M en N omzetten in een array van 11 bytes</u>
Een variabele van het type unsigned long heeft 32 bits. In de variabele N neemt de gecodeerde call 28 van deze 32 bits in. De locator en het vermogen nemen 22 bits van de variabele M in. Deze 28 + 22 = 50 bits worden achter elkaar in een array van 7 bytes geplaatst: *c[0] ...c[6].*

```
// bit packing
  for (tl1=0;tl1<3;tl1++)
      c[tl1] = (N >> (20-8*tl1)) & B11111111;
  c[3] = (N << 4) & B11110000;
  c[3] = c[3] ^ (M >> 18);
  for (tl1=4;tl1<6;tl1++)
      c[tl1] = (M >> (10-8*(tl1-4))) & B11111111;
  c[6] = (M << 6) & B11000000;
  for (tl1=7;tl1<11;tl1++) c[tl1] = 0;
```

Listing 2d: omzetten van de variabelen N en M in een array van 7 bytes.

Het verpakken van de 50 bits van *N* en *M* in de array *c* gebeurt in vijf stappen. In de eerste stap worden *c[0]* t/m *c[2]* gevuld met de eerste 24 bits uit *N* waarbij begonnen wordt bij het hoogste bit van *N*. In de tweede stap worden de vier hoogste bits van *c[3]* gevuld met de laatste 4 bits van de 28 bits van *N* die gegevens van de call bevatten. In de derde stap worden de laagste 4 bits van *c[3]* gevuld met de hoogste 4 bits van de variabele *M*. In de vierde stap worden *c[4]* en *c[5]* gevuld met de volgende 16 bits. Van de 22 bits met informatie uit de variabele *M* zijn er nu 20 in de array

c[0…5] geplaatst. In de vijfde stap worden de laatste twee bits van de 22 bits aan de hoogste bits van *c[6]* toegevoegd. Daarmee is de klus nog niet af. In de laatste regel van de listing krijgen de overige elementen (*c[7]...c[10]*) van de array de waarde 0 toegewezen. Van de in totaal 88 bits van *c[0]* t/m *c[10]* worden er in de overige stappen van de codering overigens maar 81 gebruikt.

Voor de call K1ABC, locator FN42 en vermogen 37 dBm betekent dit dat *c[0]* t/m *c[6]* de waarde 0xF7, 0x0C, 0x23, 0x8B, 0x0D, 0x19 en 0x40 hebben. Maak je zelf een programma dan kun je aan de hand van deze waarden controleren of het proces goed is verlopen.

<u>Derde stap: toevoegen voorwaartse foutcorrectie FEC</u>

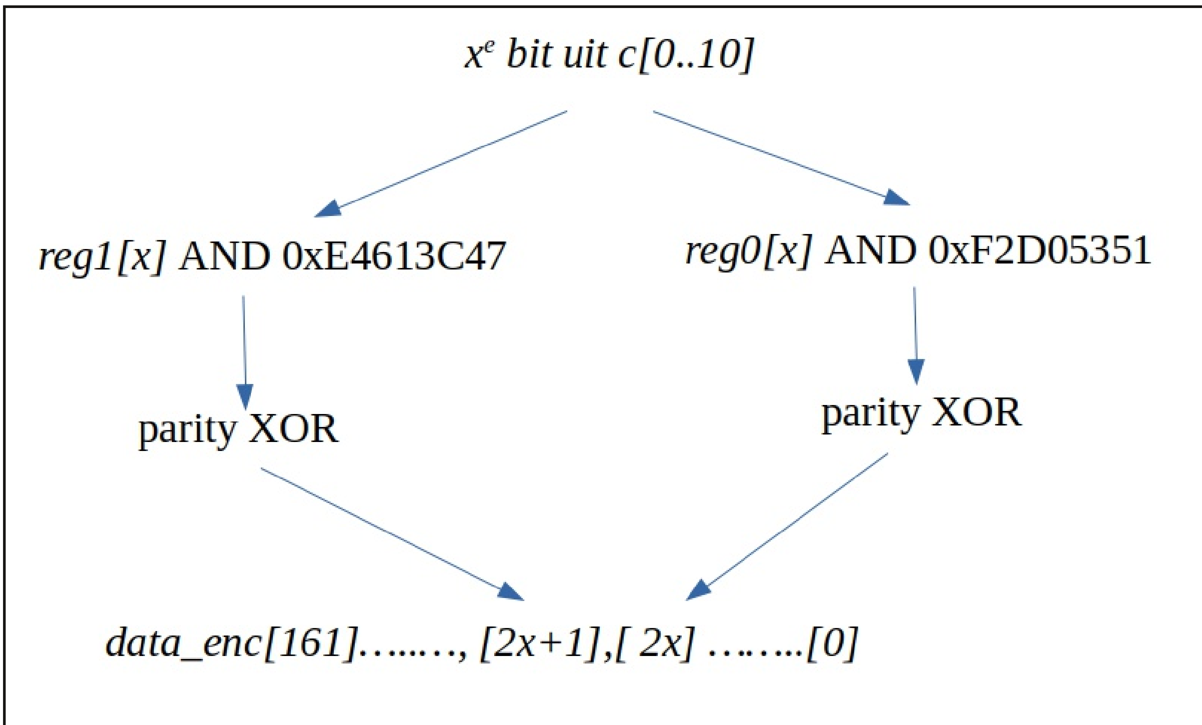
Deze stap heeft me het meeste hoofdbrekers bezorgd. Zowel voor Joe Taylor als voor Andy Talbot was FEC blijkbaar een bekend begrip maar dat was het niet voor mij. Ik heb eerst maar eens opgezocht wat FEC eigenlijk is. Daaruit bleek dat voorwaartse foutcorrectie (FEC) onder meer wordt gebruikt in situaties waarin tweeweg communicatie moeilijk of onmogelijk is. Aan de data met informatie voor de ontvanger wordt data voor controle toegevoegd op basis waarvan de ontvanger fouten in de datastroom kan herstellen. Hier zitten wel grenzen aan, bij teveel fouten kan de originele informatie niet meer hersteld worden. In de codering voor WSPR wordt gebruik gemaakt van “convolutional encoding”, een techniek gebaseerd op de bewerking van de bits met data-informatie. De 81 bits met informatie in de array *c[0...10*] worden door de FEC uitgewerkt tot een array van 162 bits. De wijze waarop dit gebeurt is voor een leek op dit gebied (ik dus...) soms moeilijk te doorgronden.

Voor het proces zijn twee registers in de vorm van unsigned long variabelen nodig: *reg0* en *reg1*. Ieder register is 32 bits lang. Het resultaat van het hele proces komt in een array van bytes: *data_enc[162]*.

Het proces start met de MSB van *c[0]* en zelf verloopt in stappen:

1. Schuif de bits in *reg0* één plaats naar links.
2. Lees de bit uit *c[0...10]* en plaats deze op de juiste plaats in *reg0*. Het MSB van *c[0]* komt op het LSB van *reg0*.
3. Bereken XOR parity van (*reg0* AND 0xF2D05351)
4. Plaats het resultaat op de opeenvolgende even plaatsen in de array, te beginnen bij *data_enc[0]*.
5. Schuif de bits in reg1 één plaats naar links.
6. Lees de bit uit *c[0...10]* en plaats deze op de juiste plaats in *reg0*. Het MSB van *c[0]* komt op het LSB van *reg1*.
7. Bereken XOR parity van (*reg0* AND 0xE4613C47)
8. Plaats het resultaat op de opeenvolgende oneven plaatsen in de array, te beginnen bij *data_enc[1]*.

In dit proces wordt ieder van de 81 betrokken bits van *c[0]* t/m *c[10]* dus verwerkt in twee bits in de resultaten array. Ik heb het proces in een schema gezet waardoor het geheel hopelijk wat duidelijker wordt:



Afbeelding 2b: toevoegen FEC aan de data

In code voor de Arduino Nano:

```
// convolutional encoding
  for (tl1=0; tl1 < 81; tl1++)
  {
     reg0=reg0<<1;
     bitWrite(reg0,0,bitRead(c[byte(tl1/8)],7-(tl1%8)));
     data_enc[tl1*2]=Parity_berekenen(reg0 & 0xF2D05351);

     reg1=reg1<<1;
     bitWrite(reg1,0,bitRead(c[byte(tl1/8)],7-(tl1%8)));
     data_enc[tl1*2+1]=Parity_berekenen(reg1 & 0xE4613C47);
  }
```

Listing 2e: toevoegen FEC aan data uit *c[0...10]*

Een belangrijk onderdeel van het proces is het berekenen van de bit parity (XOR) . Deze berekening is ondergebracht in een aparte functie:

```
byte Parity_berekenen(unsigned long data)
{
  byte tl, parity=0;

  for(tl=0; tl<32; tl++)
    parity ^= bitRead(data, tl);
  return(parity);
}
```

Listing 2f: bereken bit parity XOR

In de functie Parity_berekenen worden de 32 bits van de variabele *data* van rechts (LSB) naar links (MSB) doorlopen en in de tweede regel van de functie stuk voor stuk vergeleken (XOR) met de variabele *parity*. De variabele *parity* bewaart na iedere bit de tussenwaarde 0 of 1 en vergelijk deze met de volgende bit van de variabele. Na 32 bits komt er een resultaat in de vorm van een 0 of 1 en deze wordt door de functie retour gestuurd.

Vierde stap: opnieuw ordenen van de bits

Alle informatie als call, locator en vermogen is in de derde stap samen met de FEC achter elkaar geplaatst in een 162 bits lange array. Door interferentie, fading of andere storing kan het gebeuren dat een deel van de array tijdens de ontvangst wegvalt. Daardoor zou een blok informatie, bijvoorbeeld een letter van de locator of call, wegvallen. Wanneer de bits verspreid worden over de array valt er door de storing nog steeds een blok met bits weg maar de informatie die wegvalt zit ook nog op een andere plaats in de array die mogelijk geen last heeft van de storing. De foutcorrectie kan dan de ontbrekende informatie herstellen.

Uiteraard kun je niet random de bits opnieuw ordenen in een 162 lange array. Het ontvangende station zou dan niet weten hoe de informatie in de bits moet worden gelezen. Het systematisch herordenen van de bits verloopt als volgt:

- Er wordt een lus doorlopen van *data_enc[0]* naar *data_enc[161]* waarvoor de variabele *p* wordt gebruikt.
- Binnen de lus wordt een variabele *i* gedefinieerd die loopt van 0 t/m 254.
- De bits in de variabele *i* worden omgedraaid. 0b0000 0001 wordt dan 0b1000 000 en geplaatst in variabele *j*.
- Als *j* kleiner is dan 162 dan wordt het bit uit *data_enc[p]* geplaatst in *data_inl[j]* en worden zowel *p* als *i* met 1 verhoogd.
- Als het resultaat in *j* groter is dan 161 wordt er geen bit geplaatst en wordt de variabele *p* ook niet verhoogd. De variabele *i* wordt wel met 1 verhoogd.
- Als i de waarde 254 heeft begint i opnieuw bij 0 en als p =162 stopt de lus.

In code voor de Arduino Nano ziet dat er zo uit:

```
// interleaving

  for(tl1=0; tl1<162;tl1++) data_inl[tl1]=0xFF;

  i=0;
  p=0;
  while(p<162)
  {
      j = Bit_reverse(i);
      if(j<162)
      {
        data_inl[j]=data_enc[p];
        p++;
      }
      if (i<254) i++;
      else i = 0;
  }
```

Listing 2g: herschikken van de bits

Voor het omdraaien van de bits in de variabele i heb ik een aparte functie geschreven:

```
// draait de volgorde om van de bits van data

byte Bit_reverse (byte data)
{
  byte tl, dummy;

  for (tl=0; tl<8; tl++)
  {
    bitWrite(dummy, tl, bitRead(data, 7-tl));
  }
  return(dummy);
}
```

Listing 2h: omdraaien volgorde bits

Vijfde stap: data samenvoegen met de synchronisatie vector

In de vijfde en laatste stap van de codering worden de bits uit de array *data_inl* samengevoegd met een pseudo random synchronisatie vector tot een array *data_output* van 162 bytes met een waarde tussen de 0 en 3. De synchronisatievector is aan het einde van dit artikel weergegeven. Ieder bit van de array *data_inl* wordt samengevoegd met het corresponderende bit uit de synchronisatie vector volgens de formule:

*data_output[x] = sync_vec[x]+ 2 * data_inl[x]*

Om RAM-geheugenruimte te besparen heb ik het resultaat van deze bewerking weggeschreven in het EEPROM-geheugen van de Arduino Nano. Bij het moduleren van de zender kunnen de 162 bytes stuk voor stuk weer worden uitgelezen.

```
// merge met Sync Vector

  for (tl1=0; tl1<162; tl1++)
    {
      EEPROM.put(050+tl1, sync_vec[tl1]+ 2 * data_inl[tl1]);

      EEPROM.get(050+tl1, dummy);
      Serial.println(dummy);
    }
  Serial.println(" ");
}
```

Listing 2i: combineren met de synchronisatie vector en bewaren in het EEPROM-geheugen

Met de opdracht **EEPROM.put**() wordt het resultaat van de bewerking van de synchronisatie vector sync_vec[..] met data_inl[..] weggeschreven in de EEPROM, te beginnen bij adres 050. Om het resultaat van het hele proces te kunnen controleren zijn er nog drie regels toegevoegd. Met **EEPROM.get**() wordt de byte weer uitgelezen en via de opdracht **Serial.println**() weergegeven op het scherm van de seriële monitor die in het programma Arduino IDE is opgenomen. Deze programmaregels worden verwijderd wanneer de codering foutloos is.

En dan de hamvraag: worden de call, locator en het vermogen op de juiste wijze gecodeerd? Ik heb wel eens een amateur horen zeggen dat bij hem alles altijd ineens werkte maar helaas is dat bij mij zelden het geval. Het begrijpen van het coderingsproces en daarna omzetten in goede en foutloze programmacode was echt wel een klus met vallen en opstaan. Gelukkig heeft Joe Taylor in zijn handleiding ook de code weergegeven die je mag verwachten bij: “K1ABC FN42 37”. Daardoor

kon ik zien dat de codering van call, locator en vermogen uiteindelijk goed geprogrammeerd was.

```
Channel symbols:
      3 3 0 0 2 0 0 0 1 0 2 0 1 3 1 2 2 2 1 0 0 3 2 3 1 3 3 2 2 0
      2 0 0 0 3 2 0 1 2 3 2 2 0 0 2 2 3 2 1 1 0 2 3 3 2 1 0 2 2 1
      3 2 1 2 2 2 0 3 3 0 3 0 3 0 1 2 1 0 2 1 2 0 3 2 1 3 2 0 0 3
      3 2 3 0 3 2 2 0 3 0 2 0 2 0 1 0 2 3 0 2 1 1 1 2 3 3 0 2 3 1
      2 1 2 2 2 1 3 3 2 0 0 0 0 1 0 3 2 0 1 3 2 2 2 2 2 0 2 3 3 2
      3 2 3 3 2 0 0 3 1 2 2 2

Decoded message: K1ABC FN42 37          ntype: 37
```

Afbeelding 2c: resultaat bij een succesvolle codering

Voor de volledigheid ook nog de synchronisatie vector:

```
unsigned int sync_vec[162]=                                 // synchronisatie vector
{1,1,0,0,0,0,0,0,1,0,0,0,1,1,1,0,0,0,1,0,0,1,0,1,1,1,1,0,0,0,0,0,0,0,1,0,0,1,0,1,0,0,
  0,0,0,0,1,0,1,1,0,0,1,1,0,1,0,0,0,1,1,0,1,0,0,0,0,1,1,0,1,0,1,0,1,0,1,0,0,1,0,0,1,0,
  1,1,0,0,0,1,1,0,1,0,1,0,0,0,1,0,0,0,0,0,1,0,0,1,0,0,1,1,1,0,1,1,0,0,1,1,0,1,0,0,0,1,
  1,1,0,0,0,0,0,1,0,1,0,0,1,1,0,0,0,0,0,0,0,1,1,0,1,0,1,1,0,0,0,1,1,0,0,0};
```

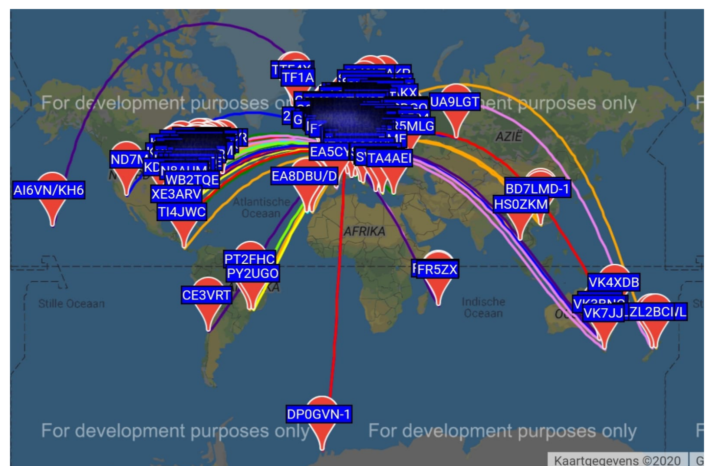
Listing 2j: de synchronisatie vector

In de derde en laatste aflevering van deze serie komt het instellen en moduleren van de oscillator en het aansluiten en programmeren met de GPS aan bod.

Ruud Jongeling
PE2BS / PE75BS

PA3CNO's Blog

In de RAZ WhatsApp groep gaat het al weken over WSPR en bijbehorende proeven met antennes, tuners en voedingslijnen. De meningen die ik daar voorbij zie komen over de zegeningen van Baluns deel ik niet volledig, maar daar ga ik in een artikel nog wel eens over uitwijden. Een WhatsAppgroep leent zich daar niet zo goed voor. Wat wel is aangetoond, is dat WSPR met 10W een volledig zinloze exercitie is. Bart PA3HEA testte het gedurende 24 uur, en werd over de hele wereld gehoord - tot op de zuidpool aan toe. Dat is geen propagatie onderzoek meer, want je krijgt zo geen beeld in welke richting je antenne het beste werkt. Misschien is dat nog enigszins af te leiden aan de gerapporteerde signaalsterkte, maar dan nog. Dit is niet waar WSPR voor bedoeld is.



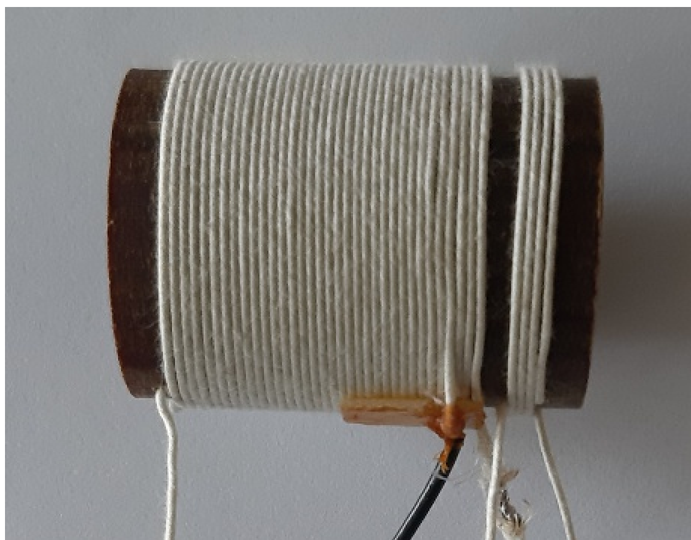
Zelf ben ik meters aan het maken met de bouw van mijn Paraset replica. Een van de klussen die ik de afgelopen tijd ter hand heb genomen, is het "verouderen" van componenten. Je loopt daarbij tegen allerlei interessante dingen aan die

vroeger heel gewoon waren. Of die nu tot het terrein van specialisten behoren. Via allerlei internet omzwervingen vond ik uit dat dat spul waarmee men vroeger de blanke draden van componenten isoleerde, oliekous heet. En dat bestaat nog steeds, al moet je met een lantaarntje zoeken naar leveranciers. Ik vond het bij bendijkman.nl, die ook te vinden is op het Nederlandse Forum voor oude radio's.

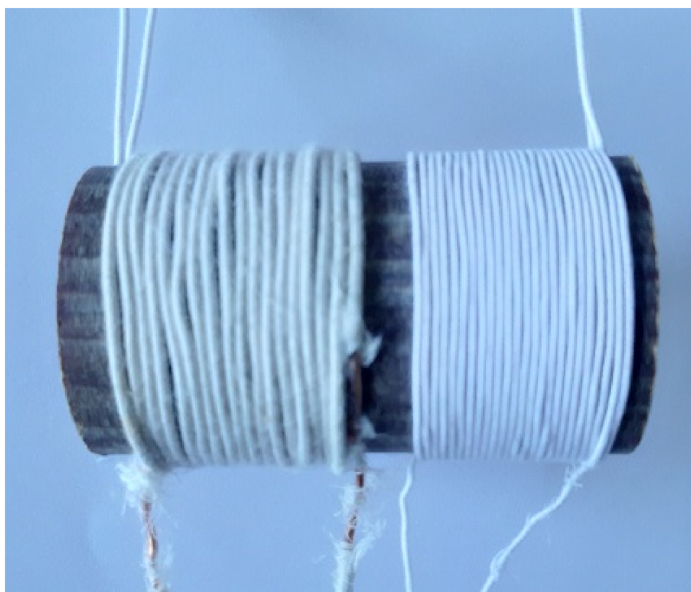


Ben heeft het in lichte en in donkere uitvoering: ik koos voor de donkere. Dat kwam het meest overeen met dat spul dat nog aan sommige oude mica condensatoren in mijn verzameling zit.

Ook het wikkelen van de spoelen leverde een uitdaging op. De originele spoelen werden gewikkeld met katoen omsponnen koperdraad. Ik zou niet weten of het in Nederland nog te koop is, maar ik vond via de links op paraset.nl een Engelse leverancier van Double Cotton Covered Copper wire. Nou is de interpretatie van de diameter best wel een dingetje. Volgens de instructies die ik had van de spoel zou deze gewikkeld moeten worden met 0,65mm katoen-draad. Op de website van wires.co.uk stond een klos van 0,71mm. Goed genoeg, dacht ik. Die 0,06mm verschil maal 35 windingen maakt aan het eind 2mm uit en dan boor ik de gaatjes wel iets verder uit elkaar, of ik druk de windingen flink aan zodat het katoen iets indeukt. Moet kunnen. Dus ik enthousiast aan het wikkelen geslagen, maar bij winding 25 ging ik al over mijn eindgat heen en toen moest ik er nog 10... Verder in elkaar drukken ging ook niet. Uiteinde-lijk de schuifmaat er maar eens bij gepakt om te zien hoe groot de afwijking nou eigenlijk was, en toen bleek de draaddiameter alleen te slaan op het koperdraad. Het katoen komt daar nog eens bij... Maar wires.co.uk had geen kleinere diameter. Uiteindelijk vond ik het bij een Engels knutselwinkelje, Vycombe-arts, En dit draad had wél een diameter van ongeveer 0,65mm dus kon ik mijn ontvangstspoel wikkelen.



Hetzelfde gold voor de spoel voor de zender: ik moest voor alles een maatje kleiner nemen. De zenderspoel heeft 4 wikkelingen: een tank spoel, een antennespoel en twee keer een winding aan de binnenkant van de spoelvorm om de indicator lampjes te voeden.



Tip voor als je ooit met dit draad aan de gang gaat: boor je doorvoergaatjes groot genoeg. Met dit draad beschadig je anders makkelijk het katoen, zoals op de foto te zien is. Niet onoverkomelijk, maar ook niet erg netjes.

De spoelen zijn daarna in de blanke vernis gezet, om te voorkomen dat het katoen vocht op gaan nemen of anderszins vies wordt.

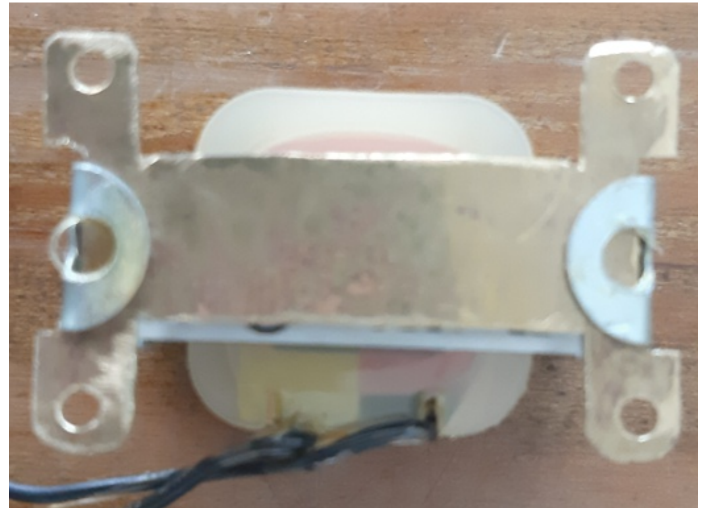
Een andere component die ik onder handen heb genomen is de smoorspoel. In het origineel zit een smoorspoel van 36H en zoek die maar eens tegenwoordig. Ik kocht (alweer bij Paraset.nl) een Hammond smoorspoel van 60H, maar dat is natuurlijk zo'n modern uitziend trafootje. Dat moest dus anders.



Let ook op die montageflenzen aan de zijkant. Mijn voorgeboorde paneel had 4 gaatjes voor de montage van de smoorspoel, niet twee. Dus zaagde ik van een messing plaat die ik nog had staan een voet die op de 4 gaatjes van het frontpaneel paste.



Let op de twee uitsparingen links en rechts. Daar kan je dan mooi die flenzen omheen buigen, en dan zit de smoorspoel vast op zijn nieuwe voetje.



De volgende fase was de twee aansluitdraden naar boven leiden en die witte isolatie van de wikkeling omwikkelen met zwart isolatieband. Dan blijven de draden zitten en tevens ziet dat zwart er beter uit dan wit. Op de originele smoorspoel zat een pertinax plaatje met twee soldeerlippen waarmee de smoorspoel aangesloten kon worden. Maar waar haal je nog een plaatje pertinax vandaan... Nou, uit de voorraad van Hugo PA2HW. Hij hielp me aan een plaatje pertinax waaruit ik een stuk zaagde voor de aansluitingen. Op het web had ik dubbele soldeerogen gevonden, en tevens had ik popnagels gekocht om ze vast te zetten. Nou durfde ik niet de popnageltang erop te zetten omdat ik bang was dat de kracht het pertinax zou splijten. Dus zaagde ik de popnagels net achter de kop af, kon het vrijkomende hulsje toen door de gaatjes drukken en met de centerpunt sloeg ik de achterkant voorzichtig uiteen zodat ze vast zaten.

Om het helemaal af te maken moest de trafo nog in de schellak gezet. Het volgende probleem. Dat koop je niet even bij de drogist. Uiteindelijk vond ik het bij dekwast.nl, een zaak in schildersartikelen. Ik kocht 100g bruine schellak, maar dan zit je met een zak schilfers. Die moet je nog oplossen in alcohol. Mijn mengsel bestond uit 100cc alcohol van 96% (drogist) en ongeveer 70g schellak schilfers die ik in een leeg honingpotje deed. Dat oplossen van die schellak duurt een paar dagen, en je moet het af en toe omschudden. Dat deed ik meestal tijdens Teams meetings met het QRL,

weliswaar buiten beeld, maar men begon zich ernstig af te vragen of ik in het beginstadium van Parkinson zat. Iets te enthousiast geschud...

Uiteindelijk was de schellak opgelost en kon ik de moderne smoorspoel van een oude look voorzien. Het resultaat mag er zijn, al zeg ik het zelf. Zie de foto hiernaast. Het is bijna niet te geloven dat dit dezelfde smoorspoel is als aan het begin van dit verhaal.

Ik ben dus weer een hoop ervaringen rijker en ik hoop dat jullie ook wat aan mijn avonturen hebben. Volgende keer neem ik jullie verder mee op mijn reis door de wereld van oude radio's en dito onderdelen.



Nog even dit. Het leuke van het invallen van de herfst is de bijbehorende condities op VHF. Vooral 's-morgens met de mistige ochtenden heb je dan temperatuur inversies waardoor je mobiele signaal veel verder draagt dan normaal. Ik weet niet hoeveel amateurs tegenwoordig nog een set in de auto hebben naast de gebruikelijke telefoon, maar je kunt dan heel leuke verbindingen maken. Op 22 september reed ik rond een uur of 8 naar Utrecht (zou ik een half jaar geleden niet in mijn hooft gehaald hebben om op die tijd te gaan rijden, maar door het verplichte thuiswerken zijn er nog steeds geen files) toen ik Frans hoorde praten over onze repeater PI3RAZ. Vermoedelijk werkte de amateur via Saint-Hubert in het Waalse deel van

België, want die zit ook op 145.750, en als je maar meer dan S5 neerzet op de ingang van PI3RAZ gaat die open, ook al heb je geen 88.5 sub-audio. Die toon is er alleen tegen zwakke signalen. Maar zou het andersom ook werken? Ik vroeg mijn APRS data op van die dag, en dan zie je inderdaad dat mijn APRS signaal vanaf de A12 ontvangen én gedecodeerd is in Goes, een afstand die ik normaal niet zou overbruggen met APRS. We gaan dus weer een leuke tijd tegemoet waarin tijdens mistige herfstochtenden verre verbindingen te maken zijn op VHF.



Afdelingsnieuws

Bij elkaar komen is er begrijpelijkerwijs nog steeds niet bij. En met de nog steeds oplopende besmettingscijfers zal dat ook wel niet snel gebeuren. Dus in dat opzicht weinig nieuws. Ik ben razend benieuwd naar het stuwmeer aan QSL-kaarten dat nu ergens bij onze QSL-manager opgeslagen moet liggen. Ook bij mij loopt het stapeltje langzaam op. Nou zijn er niet zo heel veel verbindingen waar ik tegenwoordig nog een kaart voor stuur (en welke amateur stuurt überhaupt nog kaarten, meestal alleen de special event stations - de rest gaat via eQSL of LotW) maar er liggen er toch nog wel een paar die beantwoord moeten worden, ook van SWLs. Maar dat zal allemaal moeten wachten tot óf het virus verdwenen is, óf er een vaccin is dat afdoende werkt. Tot die tijd geen bijeenkomsten, beurzen of andere evenementen die leiden tot sociale contacten waarbij besmetting kan plaatsvinden.

Nog een terugkoppeling op de Arduno radio van Gert PE0MGB van vorige maand: onze oproep om je te melden als je geïnteresseerd was in een print voor deze radio was niet aan dovemansoren gericht. Ruim 20 amateurs uit Nederland en nog een aantal van daarbuiten hebben zich gemeld. Inmiddels denken we de prijs van de print op een paar Euro te kunnen houden, en waar we ook over gesproken hebben is om de print te leveren met de SMD Si4735 er al op gesoldeerd. Hoef je dat alvast niet meer te doen. We gaan wel een avondje met een paar amateurs op 1,5m afstand en genoeg bier voor het ontsmetten een paar van die chippies erop bakken. Anyway, we kijken het nog even aan en dan zullen we een bak printen bestellen. Degenen die interesse getoond hebben worden per email op de hoogte gehouden. Had je je nog niet gemeld als geïnteresseerde maar wil je dat alsnog, stuur dan een email naar info@pi4raz.nl en dan zetten we je op de lijst. Het is geen afname verplichting, maar om ons een idee te geven hoeveel printen we moeten bestellen.